John von Neumann Institute for Computing

Parallel Simulations of Underground Flow in Porous and Fractured Media

A. Beaudoin, J.R. De Dreuzy, J. Erhel, H. Mustapha

published in

Parallel Computing:
Current & Future Issues of High-End Computing,
Proceedings of the International Conference ParCo 2005,
G.R. Joubert, W.E. Nagel, F.J. Peters, O. Plata, P. Tirado, E. Zapata (Editors),
John von Neumann Institute for Computing, Jülich,
NIC Series, Vol. **33**, ISBN 3-00-017352-8, pp. 391-398, 2006.

http://www.fz-juelich.de/nic-series/volume33

Parallel Simulations of Underground Flow in Porous and Fractured Media

A. Beaudoin[a], J.R. De Dreuzy[b], J. Erhel[a] and H. Mustapha[a]

[a]Irisa-Inria, Campus of Beaulieu, 35042 Rennes Cedex, France.

[b]Géosciences Rennes, Campus de Beaulieu, 35042 Rennes cedex, France.

Abstract

In this paper, we present a parallel software for solving linear flow equations in two kinds of subsurface media, a 2D highly heterogeneous porous medium and a 3D fracture network. Parallel computing allows us to solve very large linear systems improving the realism of simulations. For these two applications, we perform a scalability analysis of two parallel solvers : HYPRE and PSPASES. HYPRE is a parallel iterative solver based on a V-cycle multi-grid algorithm. PSPASES is a parallel direct solver based on the Cholesky factorization.

1. Introduction

The prediction of natural underground flow circulation has brought up the concern of medium heterogeneity. Geological heterogeneity in porous media occurs on a large range of scales, that goes from the mineral scale (of the order of the millimeter) to the formation scale (that can be larger than a kilometer). Similarly, rock solid masses are in general fractured and fluids can percolate through networks of interconnected fractures, which are also heterogeneous and multi-scale. Because of the difficulty in reaching the natural medium, the numerical approach seems to be the best solution to study the influence of these two kinds of heterogeneity on the underground flow circulation. The numerical simulations are obtained by performing three main phases : generation of a linear system, solution of the linear system and evaluation of the flow. The first phase is obtained by discretizing the governing equations which are the mass conservation equation and Darcy's law for steady-state, incompressible and single-phase flow in porous media. We use a Mixed Finite Element method to discretize the equations, because it conserves fluxes locally and globally, uses unstructured meshes well-suited to complex geometries and allows heterogeneous and anisotropic permeability tensors. The mesh of the 2D medium is a regular triangular grid, hence Mixed Finite Element method is equivalent to a Finite Difference method. On the other hand, the mesh of the 3D fracture network is rather complex, with interconnected 2D triangular meshes in each fracture. The discrete problem to solve is linear, with a sparse symmetric positive definite matrix. The very strong variability of hydraulic properties leads to an ill-conditioned matrix. For the second phase, we can use direct or iterative methods. The third phase is performed by evaluating Darcy's law. The numerical study of the influence of these two kinds of heterogeneity on the underground flow circulation needs to generate a large number of realistic numerical simulations leading to very large sparse linear systems. In order to reach this objective, we have to overcome two main problems : memory size to generate very large linear systems and run time to solve a large number of linear systems. High performance computing is thus mandatory in this framework. This paper is organized as follows : in Section 2, a description of the parallel algorithms used to generate and solve the large linear systems is given. In Section 3, an analysis of performances of parallel linear solvers applied to two media is presented. Finally, a short summary and our future work are presented in Section 4.

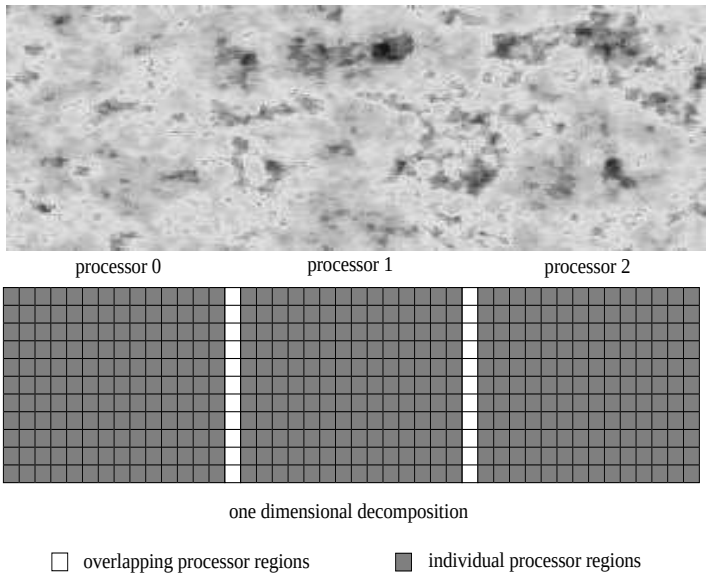

Figure 1. Example of an one-dimensional domain decomposition of 2D highly heterogeneous porous medium (top picture : heterogeneous hydraulic conductivity field and bottom picture : mesh).

2. Parallel Computing

2.1. Parallel matrix generation

In the case of 2D medium, the computational domain is a regular triangular grid on which a random hydraulic conductivity field K is generated. This random hydraulic conductivity field K follows a stationary log-normal probability distribution $Y = \ln(K)$, which is defined by a mean m_Y and a covariance function C_Y. The porous medium is assumed to be isotropic. The covariance function is then defined by $C_Y(\mathbf{r}) = {\sigma_Y}^2 \exp\left(-\frac{|\mathbf{r}|^2}{\lambda_Y}\right)$ where ${\sigma_Y}^2$ is the variance of the log hydraulic conductivity and λ_Y denotes the correlation length scale. To generate the random hydraulic field, a spectral simulation based on the FFT method (Fast Fourier Transform method) is used [11] [10]. The evaluation of Fourier transform in the FFT method has been performed with the FFTW library (Fast Fourier Transform in the West). This FFTW library allows to calculate the Fourier transform on a cluster of processors. Data is then distributed across the processors [4]. The computational grid is divided according to the columns. Thus each processor gets a subset of columns of the computational grid using a one dimensional distribution along the columns. For the flow problem, we have decided to keep this one dimensional decomposition in the y direction. Each processor generates the sub-matrix corresponding to its sub-domain. In order to evaluate the element matrices which are on the boundary of a processor domain, we include one boundary layer of ghost cells that overlaps each sub-domain. These ghost cells are used for temporary storage of grid quantities from neighboring processors. It allows to reduce the communications between the processors during the assembly of linear system (see Figure 1). The 3D fracture network is generated by using a stochastic approach. The network is included in a cube of size L, fractures are ellipses and fracture length is modeled by a random power-law distribution. Eccentricity, orientation and position are also randomly distributed. The density of the network is a parameter of the network generation [2]. The linear system is obtained by discretizing the flow equation on a global mesh of the network. In a first step, all the fracture intersections and boundaries are discretized. A 2D mesh of each fracture is then generated

by using the Emc2 software [6]. It ensures that the intersections are equally discretized in common fractures and that the flux is conserved across the fractures. A similar method is developed in [9], but with non-matching discretized intersections and an adapted nonconforming Mixed Finite Element Method. The equations are then approximated by using the mesh and lead to a linear system of equations. In order to get a symmetric positive definite matrix, a hybrid approach is used [7]. The order of the matrix is the number of edges in the network mesh. The parallel mesh generation relies on a data distribution of fracture structures. In order to get a static balanced task scheduling, we implement a variant of the bin packing algorithm [1]. The mesh generation is embarrassingly parallel, communications occur only to attribute global numbers to mesh edges. Then we infer the data distribution of the mesh and the matrix structures from the parallel mesh generation. Therefore matrix generation is done in parallel with the same distribution. All fracture intersections are processed by one unique processor, which collects matrix data related to intersections from other processors (see Figure 2).

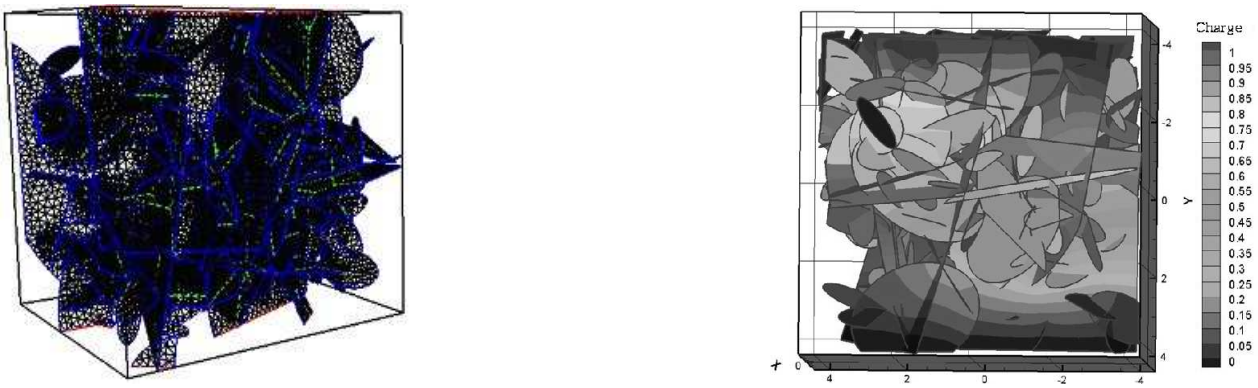

Figure 2. Example of mesh and flow computation of 3D fracture network (left picture : mesh and right picture : flow computation).

2.2. Parallel linear solving

For both 2D highly heterogeneous porous medium and 3D fracture network, the linear system obtained from the discretization of equations with the Mixed Finite Element method is characterized by a sparse symmetric positive definite matrix. For solving these systems, we have investigated both direct and iterative methods. The direct method is based on the Cholesky factorization $A = LL^T$ which is accurate and robust. We use the PSPASES solver, which is an efficient parallel sparse direct solver for symmetric positive definite matrices. Parallelism is based on a distributed-memory paradigm and communications are handled by the MPI library. A slight drawback is that the number of processors must be a power of two, and that there is no sequential version of PSPASES [5]. Because of fill-in in the Cholesky factor L, memory requirements may be a bottleneck for very large linear systems, so it may be necessary to switch to an iterative method. Preconditioned conjugate gradient can be efficient, provided the preconditioner is powerful. For the 2D medium, we have chosen a multi-grid solver which is well adapted to solve linear systems arising from finite difference, finite volume or finite element discretizations of partial differential equations on regular grids. We use the HYPRE library which contains a parallel V-cycle multi-grid algorithm called SMG (Structured Multi-Grid). As the PSPASES library, the HYPRE library uses MPI for the communications

between the processors [3]. Once the linear system has been solved, we compute the hydraulic head and the flux on each element of the computational grid. As for the hydraulic conductivity, the hydraulic head is distributed across all processors. Each processor has the value of hydraulic head on its computational sub-domain.

3. Results and performances

3.1. Tests and architecture

The study of the performance of our parallel software is realized in three tests. In the two first tests, we analyze the complexity and the scalability of the direct solver PSPASES. In the third test, we compare the direct solver PSPASES with the iterative solver HYPRE. This last test is only applied to 2D medium. All tests are performed on a SUN cluster composed of two nodes of 32 computers each. Each computer is a 2.2 Ghz AMD Opteron bi-processor with 2 Go of RAM. Inside each node, computers are interconnected by a Gigabit Ethernet Network Interface, and the two nodes are interconnected by a Gigabit Ethernet switch (CISCO 3750). The characteristic bi-processors of cluster is not used.

3.2. Complexity analysis with a direct solver

The study of complexity is performed by analyzing the CPU time and memory requirements of our two applications. The algorithm is decomposed into three phases: matrix generation, linear solving and flow computation. Figure 3 shows the CPU time of each phase, obtained with two processors, with respect to N, the size of the linear system. We take parallel run times for two processors because the parallel solver PSPASES requires at least two processors. We can observe that the most time consuming phase is the linear solver. Moreover, a complexity analysis shows that the first and third phases have a linear behavior, with a CPU time proportional to the matrix size N. For the fracture network, at least for these experiments, the factorization step is almost linear with N. In contrary, the factorization step has a nonlinear complexity, with a CPU time proportional to N^{α}, where α is about 1.5 for the 2D grid. For both applications, the main memory requirements are to store the matrices A and L. The memory requirements are measured by counting the number of non-zeros in the sparse matrices A and L (in a sparse storage compressed scheme, only nonzero coefficients are stored in 64 bit words). On Figure 4, these two numbers, noted $nz(A)$ and $nz(L)$, are reported for N ranging from $0.5e+06$ to $6.0e+06$. For 2D grids as well as for fracture networks, $nz(A)$ is roughly $5N$, whereas $nz(L)$ is roughly $5Nlog5N$. The analysis of CPU time and memory requirements show that it is necessary to use parallel algorithms to solve large linear systems.

3.3. Scalability analysis with a direct solver

On Figure 5, the total time of a single run for a linear system of size $N = 4.2e + 06$ is reported with respect to the number of processors P ranging from 2 to 64. We can observe that the total runtime is reduced as P increases for both applications. We can also notice that it is sufficient to take 32 processors for the case of 2D medium, whereas for the case of 3D fracture network, we can use effectively 16 processors. In order to understand the parallel performances of PSPASES, we analyze the scalability which refers to the capacity of the algorithm-architecture combination to effectively utilize an increasing number of processors P. The scalability can be evaluated by studying the efficiency E or the speedup S [8]. An algorithm architecture is considered scalable if the efficiency E is fixed when the size of linear system N increases proportionally to the number of processors P. The efficiency E is defined as the ratio of the serial runtime to the parallel runtime. For the linear solving phase, the sequential runtime is unknown because PSPASES uses a number of

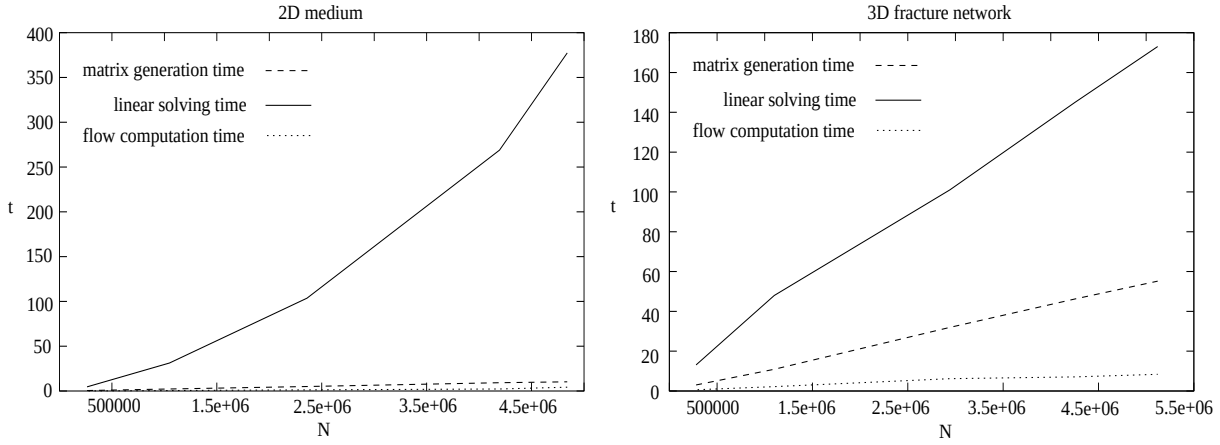

Figure 3. CPU time of three phases (matrix generation, linear solving and flow computation), obtained with two processors, with respect to the size of linear system N for both two applications.

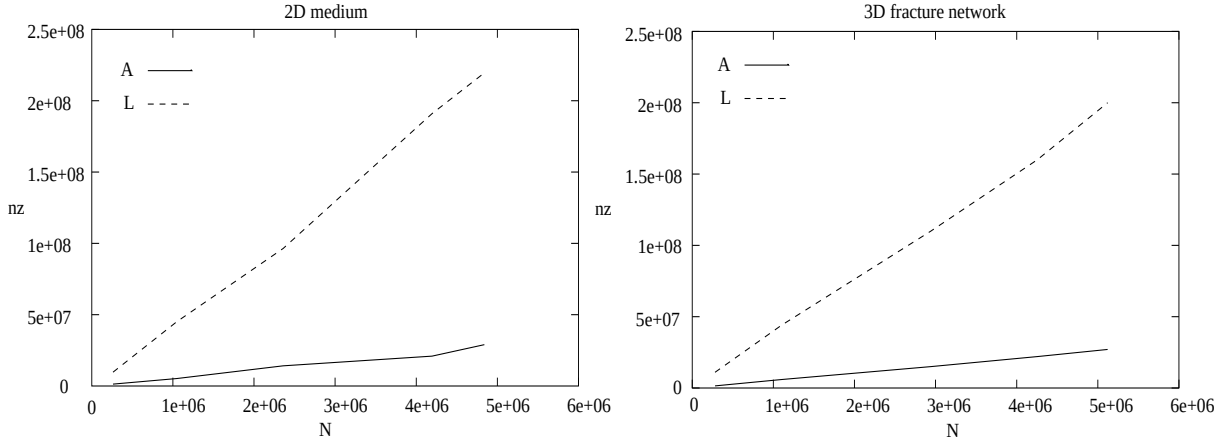

Figure 4. Number of nonzero coefficients nz in the matrices A and L with respect to the size of linear system N for both 2D medium and 3D fracture network.

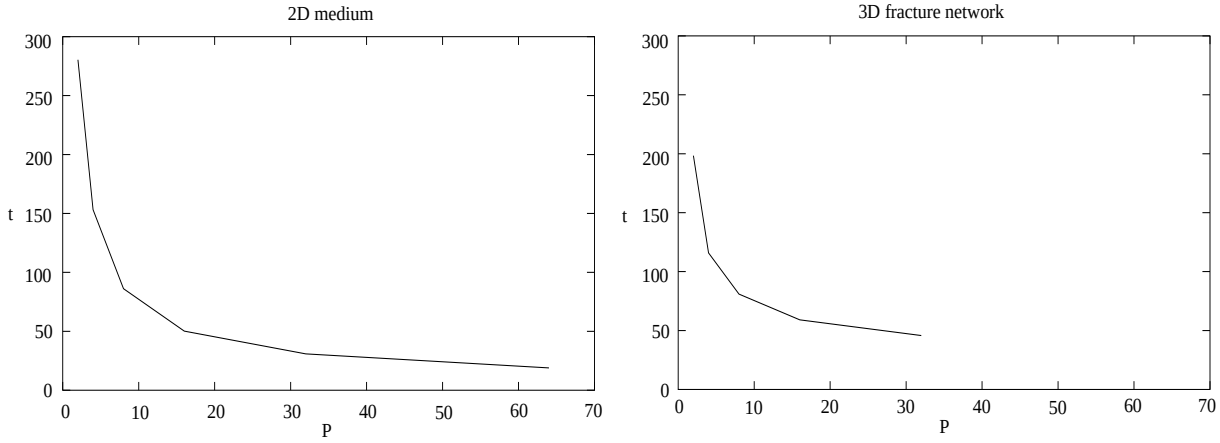

Figure 5. Total run time of a single realization for $N = 4.2e + 06$ with respect to number of processors P for both 2D medium and 3D fracture network.

P	N	T_p	R
2	262144	5.60	11977373
8	1048576	11.33	11844656
32	4194304	25.70	10443374
4	262144	2.92	11502234
16	1048576	6.06	11079774
64	4194304	13.08	10535895

P	N	T_p	R
2	262144	13.10	10006
8	1048576	22.06	5942
32	4194304	38.41	341
4	262144	7.94	16508
16	1048576	16.05	4083
64	4194304	no value	no value

Table 1
Values of the parameter R for various values of (P, N) and for both two applications (left : 2D medium and right : 3D fracture network).

processors which has to be a power of two. But we know that this sequential runtime is proportional to N^α where α is about 1.5 for the 2D grid and 1 for the 3D fracture network. The efficiency is thus proportional to $R = (N^\alpha/(PT_p))$, where T_p denotes the parallel runtime for the linear solving phase. In order to test the scalability of PSPASES, we have evaluated the parameter R by increasing the number of processors P and the size of linear system N with a coefficient equal to 4. The numerical values of R are reported in Table 1 for both applications. From these results, we can conclude that the parallel solver PSPASES is scalable in the problem of 2D medium, as predicted by the theory and observed in [5]. However, it does not appear to be scalable in the problem of 3D fracture network. For the problem of 2D medium, the efficiency E is fixed if the ratio of N to P is maintained constant. Another analysis relies on the speedup S. Since there is no sequential version of PSPASES, the speedup is given by $S = 2T_2/T_P$ where T_2 and T_p are respectively the linear solving times obtained with 2 and P processors. Figure 6 shows the speedup S with respect to the number of processors P for three values of N and for both applications. We can observe that the problem of 2D medium gives values of S larger than those of the problem of 3D fracture network and that the values of S are equivalent for three values of N in the case of 3D fracture network. These two behaviors can be explained by the total parallel overhead T_o which is defined as the total time of the overhead due to parallel processing and is equal to $PT_p - T_s$, where T_s is the serial runtime. The speedup S can then be given by $S = P/(1 + T_o/T_s)$. From [5], we can deduce that the ratio T_o/T_s is proportional to $\sqrt{P/N}$ for 2D medium. This is in good agreement with our numerical values of speedup S. We can also notice that the speedup S decreases at $P = 32$ for $N = 1e + 06$ in the case of 2D medium, probably because the total parallel overhead T_o (including reordering and triangular solving) increases faster than P. An algorithm architecture is considered scalable if the speedup S increases proportionally to the number of processors P when the problem size increases proportionally to P. For the problem of 3D fracture network, the speedup S remains constant when N increases, so that PSPASES is not scalable, probably because the ratio T_o/T_s is independent of N. In this case, parallel computing is used to speedup computations up to eight processors and mainly to increase the memory capacity with more processors.

3.4. Comparison between a direct and an iterative solver

In the case of 2D medium, the parallel direct solver PSPASES gives good performances. But the memory requirements for the matrix L increase faster than the size N of the linear system. The fill-in of matrix L can saturate available memory. As our main objective is to solve large linear systems, we use also a parallel iterative solver, called HYPRE, in order to overcome this difficulty. On Figure 7, the parallel runtime of linear solving phase has been reported with respect to the number P of processors. We can observe that PSPASES gives better performances than HYPRE for

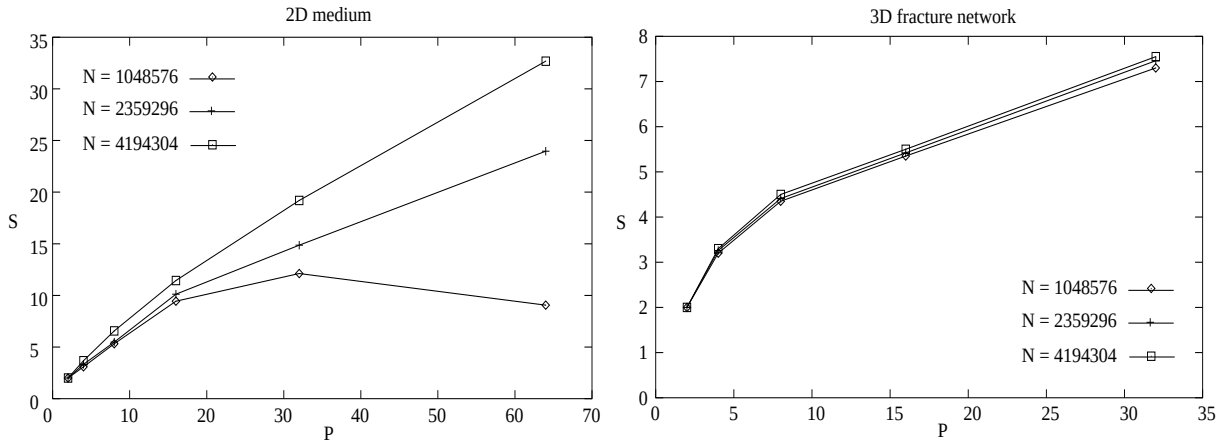

Figure 6. Speed-up S of linear solving with respect to the number of processors P for three values of N and for both 2D medium and 3D fracture network.

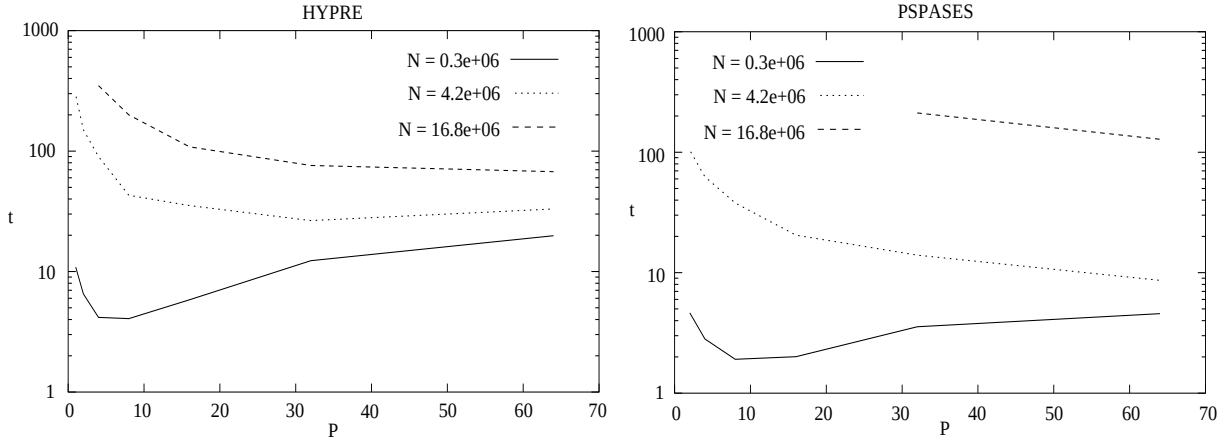

Figure 7. Linear solving time with respect to the number of processors P for N fixed and both PSPASES and HYPRE.

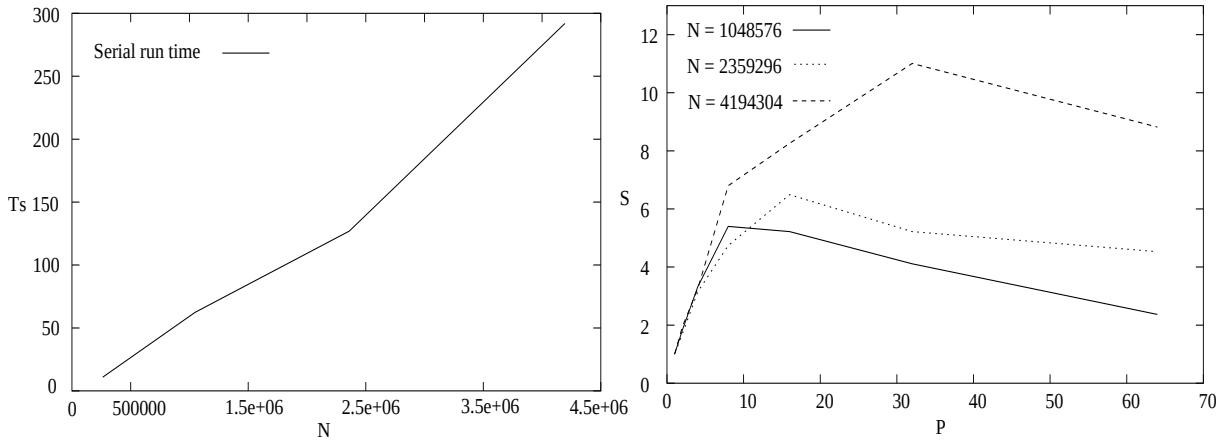

Figure 8. Serial runtime T_s of linear solving phase with respect to the size of linear system N and speedup S with respect to the number of processors P for three values of N obtained with HYPRE.

$N = 0.3e + 06$. For $N = 4.2e + 06$, the two parallel solvers give roughly the same performance. The performance of HYPRE is better than the performance of PSPASES for $N = 16.8e+06$. Figure 8 shows the serial runtime T_s of linear solving phases for values of N ranging from $0.3e + 06$ to $4.5e + 06$ and the speedup S with respect to the number P of processors for three values of N. Speedups are not as important as with PSPASES, but they increase also with N. Therefore we can conclude that HYPRE is somehow scalable for the 2D medium problem. It is necessary to pursue theoretical and practical investigations to measure the isoefficiency function.

4. Summary

In this paper, we have analyzed the performance of a parallel software that solves linear flow equations in 2D porous media or in 3D fracture networks. The scalability study shows that the parallel direct solver PSPASES is scalable in the case of 2D medium, contrary in the case of 3D fracture network. In the 3D fracture network, parallel computing improves the memory capacity, whereas it improves also the linear solving time in the case of 2D medium. However PSPASES can saturate the available memory because of fill-in of matrix L for large linear systems. In order to overcome this problem, the use of parallel iterative solver HYPRE seems to be a good solution. The comparison between the two parallel solvers, shows that HYPRE is efficient for large linear systems. In a future work, we will use HYPRE in the case of 3D fracture network. We will also realize a 3D extension of our parallel software for the case of 2D porous media. Finally, this parallel software will be used in a transport code for simulating the solute migration.

References

[1] S. Albers and M. Mitzenmacher : Average-case analyses of first fit and random fit bin packing. Random structures alg. 16, 240-259, 2000.

[2] J.R. de Dreuzy, P. Davy and O. Bour. Percolation threshold of 3D random ellipses with widely-scattered distributions of eccentricity and size, Physical Review E, vol. 62, 5948-5952, 2000.

[3] R.D. Falgout, J.E. Jones and U.M. Yang : Pursuing scalability for hypre's coceptual interfaces. ACM transactions on mathematical software, 2004.

[4] M. Frigo and S.G. Johnson : The Design and implementation of FFTW3. Proceedings of the IEEE, vol. 93, 216-231, 2005.

[5] A. Gupta, F. Gustavson, M. Joshi, G. Karypis, and V. Kumar. PSPASES : An efficient and scalable parallel sparse direct solver. In parallel numerical computations with applications, T. Yang (ed.), Kluwer international series in engineering and computer science, vol. 515, 1999.

[6] F. Hecht and E. Saltel : Emc2 : Un logiciel d'édition de maillages et de contours bidimensionnels. Rapport technique n 118, INRIA, 1990.

[7] H. Hoteit, J. Erhel, R. Mosé, B. Philippe and P. Ackerer : Numerical reliability for mixed methods applied to fbw problems in porous media. Computational geosciences 6(2): 161-194, 2002.

[8] V. Kumar and A. Gupta : Analyzing scalability of parallel algorithms and architectures. Journal of parallel and distributed computing, 1994.

[9] P.A. Raviart and J. M. Thomas : A mixed hybrid finite element method for the second order elliptic problem. in Lectures Notes in Mathematics 606: 292-315, 1977.

[10] M.G. Trefry, F.P. Ruan and D. McLaughlin : Numerical simulations of preasymptotic transport in heterogeneous porous media: Departures from the Gaussian limit. Water Resources Research, vol. 39, 1063, 2003.

[11] T. Yao : Reproduction of the mean, variance and variogram model in spectral simulation. Mahematical geology, vol. 36, 487-505, 2004.